Documentacion del código

alta

Endpoints

POST `/api/v2/materias/`

POST `/api/v1/alumnos/`

Descripción

Este endpoint permite realizar una solicitud HTTP `POST` al backend con los datos del formulario, para registrar una nueva materia en el sistema.

Solicitud

Requisitos:

- **Cuerpo de la solicitud (Request Body):**
  Debe enviarse un objeto JSON con la siguiente estructura:

```
{
  "formularioMateria": {
    "id": "form-editar",
    "descripcion": "Formulario HTML para editar materias."
  }
}
```

Esquema del Request Body:

- `formularioMateria` *(object)*: Contiene los datos procesados del formulario.
  - `id` *(string)*: Identificador del formulario. Ejemplo: `"form-editar"`.
  - `descripcion` *(string)*: Descripción del formulario. Ejemplo: `"Formulario HTML para editar materias."`.

Respuestas

200 - Éxito
La materia fue creada exitosamente.
Cuerpo de la respuesta (Response Body):

```
{
  "mensaje": "Materia creada con éxito."
}
```

400 - Error
Ocurrió un error en los datos enviados.
Cuerpo de la respuesta (Response Body):

json

Copiar código

```
{
  "mensaje": "No se pudo dar de alta la materia."
}
```

editar

Endpoints

GET `/api/v1/alumnos/{id}`

GET `/api/v2/materias/{id}`

Descripción

Obtiene los datos de un alumno específico para renderizarlos en un formulario.

Parámetros:

- `id` *(string)*: Identificador único del alumno.

Respuestas

200 - Éxito
Los datos del alumno fueron recuperados correctamente.
Cuerpo de la respuesta (Response Body):

```
{
  "id": "123",
  "nombre": "Juan Pérez",
  "edad": 20,
  "carrera": "Ingeniería en Sistemas"
}
```

404 - No encontrado
El alumno con el `id` proporcionado no existe.

PUT `/api/v1/alumnos/{id}`

PUT `/api/v2/materias/{id}`

Descripción

Actualiza la información de un alumno existente.

Parámetros:

- **`id` *(string)*: Identificador único del alumno.**

Requisitos del Cuerpo de la Solicitud:

- **Enviar un objeto JSON con los datos actualizados del alumno:**

```
{
  "nombre": "Juan Pérez",
  "edad": 21,
  "carrera": "Ingeniería en Sistemas"
}
```

Respuestas

200 - Éxito
La información del alumno fue actualizada correctamente.
Cuerpo de la respuesta (Response Body):

```
{
  "mensaje": "Datos actualizados con éxito."
}
```

400 - Error

Los datos enviados son inválidos.

DELETE `/api/v1/alumnos/{id}`

DELETE `/api/v2/materias/{id}`

Descripción

Elimina un registro de alumno específico.

Parámetros:

- `id` ***(string)***: **Identificador único del alumno.**

Respuestas

200 - Éxito
El registro fue eliminado correctamente.
Cuerpo de la respuesta (Response Body):

```
{
  "mensaje": "Registro eliminado correctamente."
}
```

404 - No encontrado

El registro no existe.

funciones

Se describen dos funciones clave que interactúan con datos de materias:

> `renderizarFormularioMaterias`: Completa un formulario con la información de una materia específica recuperada desde la API.
>
> `renderizarListadoMaterias`: Llena dinámicamente una tabla con todas las materias disponibles.

`renderizarFormularioMaterias`

Descripción

Esta función toma los datos de una materia desde una solicitud HTTP y los inserta en un formulario de edición de materias.

Parámetros

- `registros` *(Response)*: Respuesta de la solicitud HTTP (fetch) que devuelve información de una materia específica.
- `formulario` *(HTMLFormElement)*: Referencia al formulario donde se llenarán los campos.

Comportamiento

1. Se espera que la respuesta sea un JSON.
2. Si la respuesta es exitosa:
    - Se llenan los campos del formulario con la información de la materia.
    - Se accede a las propiedades `nombre` y `modulos_por_semana` de la materia.
3. Si la solicitud no es exitosa, se imprime en la consola "Materia no encontrada".
4. En caso de un error inesperado, el error se captura y se relanza.

```
await renderizarFormularioMaterias(respuesta, formulario);
```

Respuesta Esperada

En caso exitoso, la función asignará valores a:

```
formulario.nombre.value = datos[0].nombre;
formulario.modulos_por_semana.value = datos[0].modulos_por_semana;
```

`renderizarListadoMaterias`

Descripción

Renderiza dinámicamente una tabla HTML con la información de todas las materias obtenidas desde el backend.

Parámetros

- `respuesta` *(Response)*: Respuesta HTTP (fetch) que devuelve un listado de todas las materias.

Comportamiento

1. Se espera que la respuesta sea un JSON.
2. Si la solicitud es exitosa:
    - Se generan dinámicamente filas en una tabla para cada materia.
    - Cada fila contiene información básica: el nombre de la materia, el número de módulos por semana, y un enlace para editar la información.
3. En caso de que la solicitud no sea exitosa, se imprime en la consola el mensaje devuelto por el servidor.
4. Se manejan errores inesperados mediante una captura (`try-catch`) para evitar que el sistema falle.

```
await renderizarListadoMaterias(respuesta);
```

index

Este código se encarga de obtener la información de todas las materias desde el backend mediante una solicitud HTTP y mostrarlas dinámicamente en la interfaz de usuario utilizando una tabla HTML.

Se utilizan las siguientes funciones:

1. `obtenerRegistros`: Realiza una solicitud HTTP para obtener la información de las materias.

2. `renderizarListadoMaterias`: Recibe la información obtenida de la solicitud y la visualiza en una tabla HTML en la página.

modelo

Este codigo genera la lógica que interactúa directamente con la base de datos (capa de datos) para realizar operaciones CRUD sobre la información

Esta documentación detalla los endpoints para manejar operaciones con los datos de alumnos en la base de datos:

- Listado completo de alumnos.
- Obtener información de un alumno por su identificador.
- Crear un nuevo registro de alumno.
- Modificar información de un alumno.
- Eliminar un alumno de la base de datos.

Obtener todos los alumnos

- **Endpoint**: `/api/v1/alumnos /api/v2/materias`
- **Método HTTP**: `GET`
- **Descripción**: Devuelve una lista con todos los alumnos almacenados en la base de datos.
- **Respuesta esperada**:

```
[
  {
    "id": 1,
    "dni": "12345678",
    "nombre": "Juan",
    "apellido": "Perez",
    "email": "juan.perez@mail.com"
  },
  {
    "id": 2,
    "dni": "87654321",
    "nombre": "Ana",
    "apellido": "Lopez",
    "email": "ana.lopez@mail.com"
  }
]
```


Obtener un alumno por ID

- **Endpoint**: `/api/v1/alumnos/{id} /api/v2/materias/{id}`
- **Método HTTP**: `GET`
- **Parámetros**:
  - **`id`**: El identificador único del alumno.
- **Descripción**: Devuelve la información de un alumno en específico.
- **Respuesta esperada**:


```
{
  "id": 1,
  "dni": "12345678",
  "nombre": "Juan",
  "apellido": "Perez",
  "email": "juan.perez@mail.com"
}
```

Crear un nuevo alumno

- **Endpoint**: `/api/v1/alumnos/alta`
- **Método HTTP**: `POST`
- **Parámetros esperados en el body**:
    - `dni`: string
    - `nombre`: string
    - `apellido`: string
    - `email`: string
- **Descripción**: Crea un nuevo alumno en la base de datos.
- **Respuesta esperada**:

```
{
  "id": 2,
  "dni": "12345678"
}
```

Actualizar un alumno

- **Endpoint**: `/api/v1/alumnos/{id} /api/v2/materias/{id}`
- **Método HTTP**: `PUT`
- **Parámetros esperados en el body**:
    - `dni`: string
    - `nombre`: string
    - `apellido`: string
    - `email`: string
- **Descripción**: Actualiza la información de un alumno por su identificador.
- **Respuesta esperada**:

```
{
  "id": 1,
  "dni": "12345678"
}
```

Eliminar un alumno

- **Endpoint**: `/api/v1/alumnos/{id} /api/v2/materias/{id}`

- **Método HTTP**: `DELETE`
- **Parámetros**:
    - `id`: Identificador del alumno
- **Descripción**: Elimina un alumno de la base de datos.

controlador

Este código representa la lógica para responder solicitudes HTTP con datos de alumnos.

Obtener todos los alumnos

- **URL:** `/api/v1/alumnos /api/v2/materias`
- **Método HTTP:** `GET`
- **Descripción:** Devuelve la lista completa de alumnos en la base de datos.
- **Respuestas esperadas:**
    - **200 OK:** Lista de alumnos.
    - **404 Not Found:** No se encontraron alumnos.

Obtener un alumno por ID

- **URL:** `/api/v1/alumnos/{id} /api/v2/materias/{id}`
- **Método HTTP:** `GET`
- **Parámetros:**
    - `id`: Identificador del alumno.
- **Respuestas esperadas:**
    - **200 OK:** Información del alumno.
    - **404 Not Found:** Alumno no encontrado.

Crear un nuevo alumno

- **URL:** `/api/v1/alumnos/crear`
- **Método HTTP:** `POST`
- **Cuerpo esperado:**

```
{
  "dni": "12345678",
  "nombre": "Ana",
  "apellido": "Lopez",
  "email": "ana.lopez@mail.com"
}
```

Respuestas esperadas:

- **200 OK:** Confirmación de que el alumno fue creado.
-

Modificar información de un alumno

- **URL:** `/api/v1/alumnos/{id}`
- **Método HTTP:** `PUT`
- **Parámetros:**
  - `id`: Identificador único del alumno.
- **Cuerpo esperado:**

```
{
  "dni": "12345678",
  "nombre": "Juan",
  "apellido": "Perez",
  "email": "juan.perez@mail.com"
}
```

Respuestas esperadas:

- **200 OK:** Confirmación de que el alumno fue modificado.
- **400 Bad Request:** Datos incompletos.

Eliminar un alumno

- **URL:** `/api/v1/alumnos/{id}`
- **Método HTTP:** `DELETE`
- **Parámetros:**
  - `id`: Identificador único del alumno.
- **Respuestas esperadas:**
  - **200 OK:** Confirmación de que el alumno fue eliminado.

    - **404 Not Found:** Alumno no encontrado.

rutas

Este archivo establece las rutas que permiten interactuar con la lógica de negocio definida en el archivo `controlador.alumnos.mjs`. Estas rutas definen los endpoints para las operaciones CRUD relacionadas con los alumnos.

Este código es clave para establecer las rutas y la lógica de enrutamiento para las operaciones CRUD

Obtener la lista de alumnos

- **Ruta:** `/api/v1/alumnos`
- **Método HTTP:** `GET`
- **Descripción:** Devuelve una lista con todos los alumnos de la base de datos.
- **Respuestas:**
    - **200 OK:** Devuelve un array de alumnos.
    - **404 Not Found:** No se encontraron alumnos.

Crear un nuevo alumno

- **Ruta:** `/api/v1/alumnos`
- **Método HTTP:** `POST`
- **Descripción:** Crea un nuevo alumno en la base de datos.
- **Cuerpo esperado:**

```
{
  "dni": "12345678",
  "nombre": "Ana",
  "apellido": "Lopez",
  "email": "ana.lopez@mail.com"
}
```

Respuestas esperadas:

- **201 Created:** Confirmación de que el alumno fue creado correctamente.

Obtener un alumno por su ID

- **Ruta:** `/api/v1/alumnos/:id`
- **Método HTTP:** `GET`

- **Parámetro requerido:**
  - `id`: El identificador único del alumno.
- **Respuestas esperadas:**
  - **200 OK:** Información del alumno encontrada.
  - **404 Not Found:** No existe un alumno con ese identificador.

Modificar un alumno

- **Ruta:** `/api/v1/alumnos/:id`
- **Método HTTP:** `PUT`
- **Parámetros requeridos:**
  - `id`: Identificador único del alumno a modificar.
- **Cuerpo esperado:**

```
{
  "dni": "12345678",
  "nombre": "Juan",
  "apellido": "Perez",
  "email": "juan.perez@mail.com"
}
```

Respuestas esperadas:

- **200 OK:** Alumno modificado exitosamente

Eliminar un alumno

- **Ruta:** `/api/v1/alumnos/:id`
- **Método HTTP:** `DELETE`
- **Parámetro requerido:**
  - `id`: Identificador único del alumno.
- **Respuestas esperadas:**
  - **200 OK:** Confirmación de que el alumno fue eliminado exitosamente.
  - **404 Not Found:** Alumno no encontrado.

utilidades

Este código contiene funciones de utilidad esenciales para manejar operaciones relacionadas con formularios, solicitudes HTTP (API) y manipulación de parámetros de URL.

procesarFormulario

Descripción

Convierte los datos de un formulario HTML en un objeto JavaScript utilizando la clase `FormData` y `Object.fromEntries`. Esto facilita el envío de los datos en solicitudes HTTP u otras operaciones en el frontend.

Parámetros

- `formulario`: **HTMLFormElement** — El formulario HTML que se desea procesar.

Retorno

- Devuelve un objeto JavaScript con los datos del formulario.

```
const datosFormulario = procesarFormulario(document.getElementById('miFormulario'));
console.log(datosFormulario);
```

obtenerParametroId

Descripción

Obtiene el parámetro `id` desde la URL actual utilizando `URLSearchParams`. Es útil para operaciones donde el identificador del recurso es necesario, como en la edición de un registro.

Parámetros

- **Ninguno**

Retorno

- Devuelve el valor del parámetro `id` de la URL, o `null` si no existe.

```
const id = obtenerParametroId();
console.log(id);
```

altaRegistro

Descripción

Envía una solicitud HTTP para realizar una operación de alta (`POST`, `PUT`) a una ruta especificada. Convierte los datos del formulario en formato JSON y los envía con los encabezados apropiados.

Parámetros

- `ruta`: **String** — La URL a la que se enviará la solicitud.
- `metodo`: **String** — El método HTTP que se utilizará (e.g., `POST`, `PUT`).
- `datos`: **Object** — El objeto con los datos que se enviarán en el cuerpo de la solicitud.

Retorno

- Devuelve la respuesta de la solicitud `fetch`.

```
const respuesta = await altaRegistro('/api/v1/alumnos', 'POST', {
  dni: '12345',
  nombre: 'Ana',
  email: 'ana@mail.com',
});
console.log(respuesta);
```

eliminarRegistro

Descripción

Envía una solicitud `DELETE` a una ruta específica para eliminar un recurso.

Parámetros

- `ruta`: **String** — La URL a la que se enviará la solicitud DELETE.

Retorno

- Devuelve la respuesta de la solicitud.

```
const respuesta = await eliminarRegistro('/api/v1/alumnos/1');
console.log(respuesta);
```

obtenerRegistros

Descripción

Realiza una solicitud `GET` para obtener registros de la API desde una ruta específica.

Parámetros

- `ruta`: **String** — La URL a la que se enviará la solicitud `GET`.

Retorno

- Devuelve la respuesta de la solicitud.

```
const respuesta = await obtenerRegistros('/api/v1/alumnos');
console.log(respuesta);
```

módulos

Establece las rutas para distintos recursos (en este caso, **alumnos y materias**)

Importaciones:

- Se importan los módulos de rutas para las distintas versiones de la API:
  - **V1 para Alumnos**: `rutasAlumnosV1`
  - **V2 para Materias**: `rutasMateriasV2`

Configuración del Router Principal:

- Se crea un router principal (`modulosApi`) que centraliza las rutas de la aplicación.
- `modulosApi.use(rutasAlumnosV1)` y `modulosApi.use(rutasMateriasV2)` indican que todas las solicitudes que vayan a las rutas especificadas se delegarán a los routers definidos en esos módulos.

Exportación del Router Principal:

- El router principal `modulosApi` se exporta para ser utilizado en el servidor principal de Express.